Diese Kopfleiste bitte unbedingt ausfüllen!

Familienname, Vorname (bitte durch eine Leerspalte trennen)

Bereich	Berufsnummer	IHK-Nummer	Prüflingsnummer
6 7	1 2 0 1		
Sp. 1-2	Sp. 3-6	Sp. 7-9	Sp. 10-14

Termin: Mittwoch, 26. April 2023

IHK

Abschlussprüfung Sommer 2023

1201

2 Entwicklung und Umsetzung von Algorithmen

Fachinformatiker
Fachinformatikerin
Anwendungsentwicklung (AO 2020)

Teil 2 der Abschlussprüfung

4 Aufgaben
mit Belegsatz
90 Minuten Prüfungszeit
100 Punkte

Bearbeitungshinweise

1. Bevor Sie mit der Bearbeitung der Aufgaben beginnen, überprüfen Sie bitte die **Vollständigkeit** dieses Aufgabensatzes. Die Anzahl der zu bearbeitenden Aufgaben ist auf dem Deckblatt links angegeben. Wenden Sie sich bei Unstimmigkeiten sofort an die Aufsicht, weil Reklamationen am Ende der Prüfung nicht anerkannt werden können.
2. Füllen Sie zuerst die **Kopfzeile** aus. Tragen Sie Ihren Familiennamen, Ihren Vornamen und Ihre Prüflings-Nr. in die oben stehenden Felder ein.
3. Lesen Sie bitte den **Text** der Aufgaben ganz durch, bevor Sie mit der Bearbeitung beginnen.
4. Halten Sie sich bei der Bearbeitung der Aufgaben genau an die **Vorgaben der Aufgabenstellung** zum Umfang der Lösung. Wenn z. B. vier Angaben gefordert werden und Sie sechs Angaben anführen, werden nur die ersten vier Angaben bewertet.
5. Tragen Sie die frei zu formulierenden **Antworten dieser offenen Aufgaben** in die dafür lt. Aufgabenstellung vorgesehenen Bereiche (Lösungszeilen, Formulare, Tabellen u. a.) des Arbeitsbogens ein.
6. Sofern nicht ausdrücklich ein Brief oder eine Formulierung in ganzen Sätzen gefordert werden, ist eine **stichwortartige Beantwortung** zulässig.
7. Schreiben Sie deutlich und gut lesbar. Ein nicht eindeutig zuzuordnendes oder **unleserliches Ergebnis** wird als **falsch** gewertet.
8. Zur Lösung der Rechenaufgaben darf ein nicht programmierter, netzunabhängiger **Taschenrechner** ohne Kommunikationsmöglichkeit mit Dritten verwendet werden.
9. Wenn Sie ein **gerundetes Ergebnis** eintragen und damit weiterrechnen müssen, rechnen Sie (auch im Taschenrechner) nur mit diesem gerundeten Ergebnis weiter.
10. Für **Hilfsaufzeichnungen** können Sie das in der Tasche beigelegte Konzeptpapier verwenden. Bewertet werden jedoch grundsätzlich nur Ihre Eintragungen in diesem Aufgabensatz.

Wird vom Korrektor ausgefüllt!

Bewertung

Für die Bewertung gilt die Vorgabe der Punkte in den Lösungshinweisen.

1. Aufg. ☐☐ (15 16) Punkte 2. Aufg. ☐☐ (17 18) Punkte 3. Aufg. ☐☐ (19 20) Punkte 4. Aufg. ☐☐ (21 22) Punkte

Prüfungszeit ☐ (23)

Die entsprechende Ziffer (1, 2 oder 3) finden Sie in der Abfrage nach der Prüfungszeit im Anschluss an die letzte Aufgabe.

Gesamtpunktzahl ☐☐☐ (24 25 26)

Prüfungsort, Datum

Unterschrift

Gemeinsame Prüfungsaufgaben der Industrie- und Handelskammern. Dieser Aufgabensatz wurde von einem überregionalen Ausschuss, der entsprechend § 40 Berufsbildungsgesetz zusammengesetzt ist, beschlossen. Hinweis: Im Interesse einer besseren Lesbarkeit wird in der Aufgabenstellung und in den Angaben zur Aufgabenstellung nur die männliche Form (generisches Maskulinum) verwendet. Die verkürzte Sprachform beinhaltet keine Wertung und die gewählten männlichen Formulierungen gelten uneingeschränkt auch für die weiteren Geschlechter.
Die Vervielfältigung, Verbreitung und öffentliche Wiedergabe der Prüfungsaufgaben und Lösungen ist nicht gestattet. Zuwiderhandlungen werden zivil- und strafrechtlich (§§ 97 ff., 106 ff. UrhG) verfolgt. – © ZPA Nord-West 2023 – Alle Rechte vorbehalten!

Korrekturrand

Die Aufgaben 1 bis 4 beziehen sich auf die folgende Ausgangssituation:

Die Parkanlage „Wilder Garten" in Köln konnte u. a. mit ihrer reichhaltigen Pflanzenvielfalt, ihren schönen verwinkelten Wegen, ihren vielen schattigen Plätzen und ihren beliebten Attraktionen in der letzten Saison ihre Anziehungskraft als Naherholungsfläche für viele Besucher steigern. Besucher aus allen Altersgruppen und auch Reiseveranstalter zählen zu den Kunden der Parkanlage.

1. Aufgabe (25 Punkte)

Sie werden von Ihrem Ausbilder beauftragt, den Prozess zur Anlage neuer Pflanzungen mittels eines UML-Aktivitätsdiagramms zu dokumentieren. Um den Prozess zu verstehen, sprechen Sie mit mehreren Mitarbeitern.

Sie beginnen mit einem Mitarbeiter der Gartenplanung.

Dieser teilt Ihnen mit, dass er den Gärtnern den Auftrag zu einer neuen Pflanzung erteilt.

Wenn die Gärtner fertig gepflanzt haben, fertigt er Bilder des neuen Bereichs an. Wenn die Bilder gemacht sind und der Einkauf die Rechnung bezahlt hat, ist der Vorgang abgeschlossen.

Informationen eines Gärtners:

Wenn der Gärtner einen neuen Pflanzauftrag vom Gartenplaner erhält, stellt er zuerst die Anforderungen zusammen und sagt dem Einkauf welche Pflanzen zu bestellen sind.

Wenn Pflanzen ankommen, bekommt er diese vom Einkauf und bringt die Pflanzen in die Quarantänestation. Der Gärtner prüft täglich die Pflanzen und kontrolliert das Quarantäneende.

Ist eine Pflanze in Quarantäne erkrankt, so muss diese behandelt und die Quarantäne für alle Pflanzen verlängert werden.

Sind alle Pflanzen gesund und die Quarantäne abgelaufen pflanzt der Gärtner die Pflanzen.

Nach Abschluss der Pflanzarbeiten macht die Gartenplanung noch Bilder.

Der Einkauf gibt Ihnen folgende Hinweise:

Der Gärtner teilt uns mit was er braucht, dann bestellen wir diese Pflanzen bei unserem Lieferanten. Bei Anlieferung übergeben wir die Pflanzen an den Gärtner und bezahlen die Rechnung.

Weiterhin beschreibt der Einkauf den Vorgang beim Lieferanten so, dass der Lieferant den Auftrag annimmt, danach die Lieferung zusammenstellt und dann die Pflanzen ausliefert.

Vervollständigen Sie aufgrund der erhaltenen Informationen das abgebildete Aktivitätsdiagramm.

Gartenplanung	Gärtner	Einkauf	Lieferant

Korrekturrand

2. Aufgabe (25 Punkte)

Korrekturrand

Die Verwaltung des Parks möchte eine Auswertung über die Besucherzahlen für alle Tage eines vorgegebenen Monats erhalten.

Für die Besuchertickets werden daher an den Ein – und Ausgängen folgende Informationen festgehalten:
- Das Ticket wird beim Einlass gescannt und damit das Datum, die Uhrzeit und die Anzahl der Personen (Gruppentickets) gespeichert.
- Beim Hinausgehen wird jede Person mit Datum und Uhrzeit einzeln erfasst.

Der Park hat aktuell die Öffnungszeiten von 9:00 Uhr bis 19:00 Uhr.

Zur Auswertung soll für jeden Tag eines Monats festgehalten werden, wie viele Personen sich von 9:00 Uhr bis 9:59 Uhr, von 10:00 Uhr bis 10:59 Uhr, … und von 18:00 Uhr bis 18:59 Uhr aufgehalten haben.

Die folgende Klasse ist bereits vorhanden:

ComeLeave
– date: Date – comeInOut: Integer – noPeople: Integer …

Für jedes Attribut sind öffentliche Zugriffsmethoden (set/get) vorhanden.

Folgende Methoden können verwendet werden:

getDay() der Klasse Date	Ermittelt zum Datum den Tag (1 – 31)
getHour() der Klasse Date	Ermittelt zum Datum die Stunde (0 – 23)
getDaysOfMonth() der Klasse Date	Ermittelt zum Datum die Anzahl der Tage im Monat

In der Tabelle **entry** vom Typ **ComeLeave** sind die Kommen – und Gehenzeiten der Besucher für den auszuwertenden Monat erfasst (eine Zeile entspricht den Attributen eines Objektes vom Typ ComeLeave):

date	**time**	**comeInOut** **0 = In/1 = Out**	**noPeople**
…			
22.5.2023	9:00	0	1
22.5.2023	9:00	0	2
22.5.2023	9:01	0	30
…			
22.5.2023	12:00	1	1
22.5.2023	12:01	0	2
22.5.2023	12:02	1	1
…			

Erstellen Sie eine Funktion countVisitors(entry: ComeLeave) : Integer[][], die folgende Anforderung erfüllt:

- Für jede Stunde eines Tages (Öffnungszeiten!) und für jeden Tag des auszuwertenden Monats soll die Anzahl der anwesenden Personen in einem zweidimensionalen Array vom Typ Integer gespeichert werden.

  Für jeden Tag des Monats soll eine Zeile und für jede Stunde innerhalb der Öffnungszeit eine Spalte verwendet werden.

  Dieses Array ist der Rückgabewert der Funktion countVisitors().

  Dabei wird eine Person ab einem Stundenabschnitt gezählt, wenn sie in diesem Abschnitt den Park betreten hat. Wenn eine Person den Park verlässt, wird die Anzahl erst für die nachfolgenden Abschnitte entsprechend korrigiert.

Beispielarray für ein Ergebnis:

Tag\Std	9:00-9:59	10:00-10:59	11:00-11:59	12:00-12:59	13:00-13:59	14:00-14:59	15:00-15:59	16:00-16:59	17:00-17:59	18:00-18:59
1	23	40	56	120	145	160	140	95	86	45
2	27	…	…	…	…	…	…	…	…	
…	…	…	…	…	…	…	…	…	…	
31	…	…	…	…	…	…	…	…	…	

<

countVisitors(entry: ComeLeave) : Integer[][]

Korrekturrand

3. Aufgabe (25 Punkte)

Korrekturrand

Die Verwaltung des Parks hat eine Jahresstatistik für die Nutzung der Monatstickets in Auftrag gegeben. Die implementierte Methode soll einem Test unterzogen werden.

Die Daten liegen in einer Liste, aufsteigend sortiert nach Monat, vor und haben die folgende Struktur:

```
struct TicketData
{       int Id;
        string Monat;
        string Jahr;
        int NutzungsZaehler;
}
```

Für den Test sollen die folgenden Testdaten verwendet werden:

Id	Monat	Jahr	Nutzungszaehler
0	Januar	2023	31
1	Januar	2023	31
2	Februar	2023	28

Aus diesen Daten soll die folgende Ausgabe generiert werden:

```
Nutzungsstatistik Monatstickets

Auswertung für Monat   Januar
 Minimale Nutzung:     31
 Maximale Nutzung:     31
 Durchschnitt:         31
 Gesamtanzahl Tickets: 2

Auswertung für Monat   Februar
 Minimale Nutzung:     28
 Maximale Nutzung:     28
 Durchschnitt:         28
 Gesamtanzahl Tickets: 1

Auswertung für Jahr 2023
 Minimale Nutzung:     28
 Maximale Nutzung:     31
 Durchschnitt:         30
 Gesamtanzahl Tickets: 3
```

Dieses Blatt kann an der Perforation aus dem Aufgabensatz herausgetrennt werden!

Abbildung zur 3. Aufgabe

```
01 TicketStatistik(List<TicketData> TicketDataList) : void
02  WriteLn("Nutzungssstatistik Monatstickets");
03  WriteLn();
04  TicketData td0 = TicketDataList[0];
05  string monat = td0.Monat;
06  int monatMin = GetMonatstage(monat);
07  int monatMax = 0;
08  int monatTicketZaehler = 0;
09  int monatNutzungsZaehler = 0;
10  int jahrMin = 31;
11  int jahrMax = 0;
12  int jahrTicketZaehler = 0;
13  int jahrNutzungsZaehler = 0;
14  foreach(TicketData td in TicketDataList)
15    if ( monat != td.Monat )
16      //Monatsauswertung:
17      WriteLn("Auswertung für Monat" + monat);
18      WriteLn(" Minimale Nutzung:" + monatMin);
19      WriteLn(" Maximale Nutzung:" + monatMax);
20      WriteLn(" Durchschnitt:" + monatNutzungsZaehler/monatTicketZaehler);
21      WriteLn(" Gesamtanzahl Tickets:" + monatTicketZaehler);
22      WriteLn();
23      //Monatsdaten für Jahresauswertung übernehmen:
24      jahrTicketZaehler = jahrTicketZaehler + monatTicketZaehler;
25      jahrNutzungsZaehler = jahrNutzungsZaehler + monatNutzungsZaehler;
26      if ( jahrMax < monatMax )
27        jahrMax = monatMax;
28      end if
29      if ( jahrMin > monatMin )
30        jahrMin = monatMin;
31      end if
32      //Reset Monat:
33      monatTicketZaehler = 0;
34      monatNutzungsZaehler = 0;
35      monatMax = 0;
36      monat = td.Monat;
37      monatMin = GetMonatstage(monat);
38    end if
39    //Ticketauswertung:
40    monatTicketZaehler += 1;
41    monatNutzungsZaehler += td.NutzungsZaehler;
42    if ( monatMax < td.NutzungsZaehler )
43      monatMax = td.NutzungsZaehler;
44    end if
45    if ( monatMin > td.NutzungsZaehler )
46      monatMin = td.NutzungsZaehler;
47    end if
48  end foreach
49  //Jahresauswertung:
50  WriteLn("Auswertung für Jahr" + td0.Jahr);
51  WriteLn(" Minimale Nutzung:" + jahrMin);
52  WriteLn(" Maximale Nutzung:" + jahrMax);
53  WriteLn(" Durchschnitt:" + jahrNutzungsZaehler / monatTicketZaehler);
54  WriteLn(" Gesamtanzahl Tickets" + jahrTicketZaehler);
55  WriteLn();
56 end TicketStatistik
```

a) Überprüfen Sie den Code auf der perforierten Seite 7 mit Hilfe der auf Seite 6 genannten Testdaten. Verwenden Sie dazu die Datenverlaufstabelle.

Korrekturrand

Datenverlaufstabelle:

Typ	Variable/Ausdruck	Werteverlauf
string	monat	
int	monatMin	
int	monatMax	
int	monatTicketZaehler	
int	monatNutzungsZaehler	
int	jahrMin	
int	jahrMax	
int	jahrTicketZaehler	
int	jahrNutzungsZaehler	
TicketData	td	

aa) Nennen Sie kurz zwei gefundene Fehler. 2 Punkte

ab) Machen Sie Vorschläge für die Änderung bzw. Erweiterung des Codes, so dass die erwartete Ausgabe erzeugt wird. 13 Punkte

Fortsetzung 3. Aufgabe →

b) Der Test soll alle Anweisungen mindestens einmal durchlaufen (Anweisungsüberdeckung, coverage test).

ba) Erläutern Sie die Anweisungsüberdeckung. 2 Punkte

bb) Mit den vorgegebenen Testdaten werden nicht alle Anweisungen des Codes durchlaufen.
Nennen Sie die Anweisungen, die nicht durchlaufen werden. 4 Punkte

bc) Passen Sie die Testdaten so an, dass alle Anweisungen durchlaufen werden und begründen Sie ihre Veränderungen. 4 Punkte

Dieses Blatt kann an der Perforation aus dem Aufgabensatz herausgetrennt werden!

Abbildung zur 3. Aufgabe

```
01 TicketStatistik(List<TicketData> TicketDataList) : void
02  WriteLn("Nutzungssstatistik Monatstickets");
03  WriteLn();
04  TicketData td0 = TicketDataList[0];
05  string monat = td0.Monat;
06  int monatMin = GetMonatstage(monat);
07  int monatMax = 0;
08  int monatTicketZaehler = 0;
09  int monatNutzungsZaehler = 0;
10  int jahrMin = 31;
11  int jahrMax = 0;
12  int jahrTicketZaehler = 0;
13  int jahrNutzungsZaehler = 0;
14  foreach(TicketData td in TicketDataList)
15    if ( monat != td.Monat )
16      //Monatsauswertung:
17      WriteLn("Auswertung für Monat" + monat);
18      WriteLn(" Minimale Nutzung:" + monatMin);
19      WriteLn(" Maximale Nutzung:" + monatMax);
20      WriteLn(" Durchschnitt:" + monatNutzungsZaehler/monatTicketZaehler);
21      WriteLn(" Gesamtanzahl Tickets:" + monatTicketZaehler);
22      WriteLn();
23      //Monatsdaten für Jahresauswertung übernehmen:
24      jahrTicketZaehler = jahrTicketZaehler + monatTicketZaehler;
25      jahrNutzungsZaehler = jahrNutzungsZaehler + monatNutzungsZaehler;
26      if ( jahrMax < monatMax )
27        jahrMax = monatMax;
28      end if
29      if ( jahrMin > monatMin )
30        jahrMin = monatMin;
31      end if
32      //Reset Monat:
33      monatTicketZaehler = 0;
34      monatNutzungsZaehler = 0;
35      monatMax = 0;
36      monat = td.Monat;
37      monatMin = GetMonatstage(monat);
38    end if
39    //Ticketauswertung:
40    monatTicketZaehler += 1;
41    monatNutzungsZaehler += td.NutzungsZaehler;
42    if ( monatMax < td.NutzungsZaehler )
43      monatMax = td.NutzungsZaehler;
44    end if
45    if ( monatMin > td.NutzungsZaehler )
46      monatMin = td.NutzungsZaehler;
47    end if
48  end foreach
49  //Jahresauswertung:
50  WriteLn("Auswertung für Jahr" + td0.Jahr);
51  WriteLn(" Minimale Nutzung:" + jahrMin);
52  WriteLn(" Maximale Nutzung:" + jahrMax);
53  WriteLn(" Durchschnitt:" + jahrNutzungsZaehler / monatTicketZaehler);
54  WriteLn(" Gesamtanzahl Tickets" + jahrTicketZaehler);
55  WriteLn();
56 end TicketStatistik
```

4. Aufgabe (25 Punkte)

Korrekturrand

Es liegt nachfolgender Datenbankauszug zur Verwaltung der Pflegearbeiten im botanischen Garten vor. Für die Verwaltung und Auswertung der Daten soll die Datenbankabfragesprache SQL zum Einsatz kommen.

Objekt

OID	Bezeichnung
1	Rasenfläche Süd
2	Rosengarten
3	Tropenhaus

Mitarbeiter

MID	Name	Vorname
1	Rose	Ines
2	Baum	Jürgen
31	Holz	Walter

Taetigkeit

TID	Bezeichnung
1	Rosenschnitt
2	Rasen mähen
3	Unkraut jäten

Pflegearbeit

PID	OID	TID	MID_Soll	MID_Ist	Datum_Soll	Datum_Abschluss
1	1	2	31	31	20.04.2023	20.04.2023
2	1	2	2	2	11.05.2023	NULL
3	2	1	1	2	20.03.2023	21.03.2023
4	2	1	1	NULL	01.08.2023	NULL
5	2	3	31	NULL	28.07.2023	NULL

a) Erstellen Sie eine SQL-Abfrage, mit der die Anzahl der Pflegearbeiten ermittelt wird, welche im Jahr 2023 geplant aber noch nicht abgeschlossen sind.

5 Punkte

b) Erstellen Sie eine SQL-Abfrage, mit der Sie ermitteln, an welchen Wochentagen in der Zeit vom 19.06.2023 bis einschließlich 30.06.2023 welche Pflegearbeiten in der „Außenanlage Nord" stattfinden sollen. Das Ergebnis soll nach dem Datum aufsteigend sortiert angezeigt werden.

8 Punkte

Beispiel-Ergebnistabelle:

Datum	Wochentag	Tätigkeit
19.06.2023	Montag	Heckenschnitt
19.06.2023	Montag	Rasen mähen
22.06.2023	Donnerstag	Unkraut jäten
26.06.2023	Montag	Rosenschnitt
30.06.2023	Freitag	Rosenschnitt

Fortsetzung 4. Aufgabe →

Fortsetzung 4. Aufgabe

Korrekturrand

c) Erstellen Sie eine SQL-Abfrage mit der Sie die durchschnittliche Anzahl der Tätigkeiten pro Monat im Jahr 2021 von jedem Mitarbeiter auflisten. 8 Punkte

Beispiel-Ergebnistabelle:

MID	Name	Vorname	Durchschnitt
1	Rose	Ines	25,3
2	Baum	Jürgen	17,8
19	Knoll	Jana	0
31	Holz	Walter	30,2

d) Geben Sie alle SQL-Anweisungen an, welche notwendig sind, um einen neuen Nutzer „Maier" mit dem Passwort „5jk2T?" zu erstellen und diesem die Leserechte an der Tabelle Objekt zuzuweisen. 4 Punkte

PRÜFUNGSZEIT – NICHT BESTANDTEIL DER PRÜFUNG!

Wie beurteilen Sie nach der Bearbeitung der Aufgaben die zur Verfügung stehende Prüfungszeit?

[1] Sie hätte kürzer sein können.
[2] Sie war angemessen.
[3] Sie hätte länger sein müssen.

Abschlussprüfung Sommer 2023
Lösungshinweise

Fachinformatiker/Fachinformatikerin
Anwendungsentwicklung (AO 2020)
1201

2 Entwicklung und Umsetzung von Algorithmen

Teil 2 der Abschlussprüfung

Allgemeine Korrekturhinweise

Die Lösungs- und Bewertungshinweise zu den einzelnen Handlungsschritten sind als Korrekturhilfen zu verstehen und erheben nicht in jedem Fall Anspruch auf Vollständigkeit und Ausschließlichkeit. Neben hier beispielhaft angeführten Lösungsmöglichkeiten sind auch andere sach- und fachgerechte Lösungsalternativen bzw. Darstellungsformen mit der vorgesehenen Punktzahl zu bewerten. Der Bewertungsspielraum des Korrektors (z. B. hinsichtlich der Berücksichtigung regionaler oder branchenspezifischer Gegebenheiten) bleibt unberührt.

Zu beachten ist die unterschiedliche Dimension der Aufgabenstellung (nennen – erklären – beschreiben – erläutern usw.).

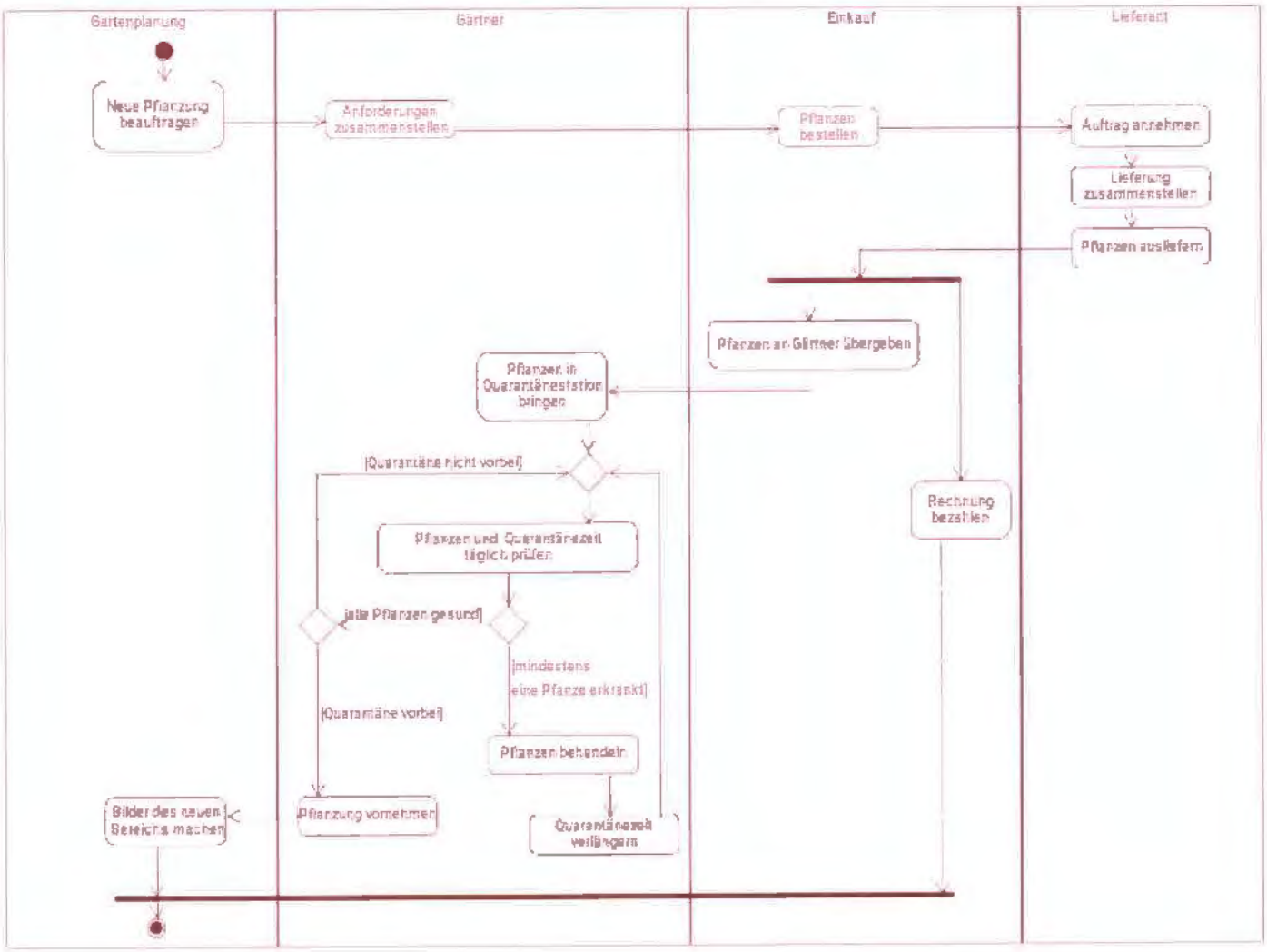
Für die Bewertung gilt folgender Punkte-Noten-Schlüssel:

Note 1	=	100 – 92 Punkte	Note 2	= unter	92 – 81 Punkte
Note 3	= unter	81 – 67 Punkte	Note 4	= unter	67 – 50 Punkte
Note 5	= unter	50 – 30 Punkte	Note 6	= unter	30 – 0 Punkte

Die Vervielfältigung, Verbreitung und öffentliche Wiedergabe der Prüfungsaufgaben und Lösungen ist nicht gestattet. Zuwiderhandlungen werden zivil- und strafrechtlich (§§ 97 ff., 106 ff. UrhG) verfolgt. – © ZPA Nord-West 2023 – Alle Rechte vorbehalten!

1. Aufgabe (25 Punkte)

Andere sinnvolle Lösungen sind als richtig zu bewerten.



2. Aufgabe (25 Punkte)

```
countVisitors(entry: ComeLeave) : Integer[][]

    count_entry = Länge von entry
    count_visitor = entry[0].getDate().getDaysOfMonth() * 10
    visitors = new Integer[count_visitor]

    for i = 0 to count_entry – 1

        day = entry[i].getDate().getDay()
        hour = entry[i].getDate().getHour()
        index_row = day – 1
        index_column = hour – 9

        if entry[i].getComeInOut() == 0
            for j = index_column bis 9
                visitors[index_row][j] = visitors[index_row][j] + entry[i].getNoPeople()
            end for
        else
            für j = index_column + 1 to 9
                visitors[index_row][j] = visitors[index_row][j] - 1
            end for
        end if

    end for

    Return visitors

end countVisitors
```

3. Aufgabe (25 Punkte)

Andere Lösungen sind möglich und entsprechend als richtig zu werten.

a) **Ausgabe des Originalcodes mit den vorgegebenen Testdaten:**

```
Nutzungsstatistik Monatstickets

Auswertung für Monat   Januar
 Minimale Nutzung:     31
 Maximale Nutzung:     31
 Durchschnitt:         31
 Gesamtanzahl Tickets: 2

Auswertung für Jahr 2023
 Minimale Nutzung:     31
 Maximale Nutzung:     31
 Durchschnitt:         62
 Gesamtanzahl Tickets: 2
```

aa) 2 Punkte

1. Februar fehlt (letzter Monat)
2. Jahresdurchschnitt falsch berechnet

ab) 13 Punkte

Korrekter Code: Die möglichen Korrekturen sind **fett** dargestellt

```
01 TicketStatistik(List<TicketData> TicketDataList) : void
02  WriteLn("Nutzungsstatistik Monatstickets");
03  WriteLn();
04  TicketData td0 = TicketDataList[0];
05  string monat = td0.Monat;
06  int monatMin = GetMonatstage(monat);
07  int monatMax = 0;
08  int monatTicketZaehler = 0;
09  int monatNutzungsZaehler = 0;
10  int jahrMin = 31;
11  int jahrMax = 0;
12  int jahrTicketZaehler = 0;
13  int jahrNutzungsZaehler = 0;
14  foreach(TicketData td in TicketDataList)
15    if ( monat != td.Monat )
16      //Monatsauswertung:
17      WriteLn("Auswertung für Monat" + monat);
18      WriteLn(" Minimale Nutzung:" + monatMin);
19      WriteLn(" Maximale Nutzung:" + monatMax);
20      WriteLn(" Durchschnitt:" + monatNutzungsZaehler/monatTicketZaehler);
21      WriteLn(" Gesamtanzahl Tickets:" + monatTicketZaehler);
22      WriteLn();
23      //Monatsdaten für Jahresauswertung übernehmen:
24      jahrTicketZaehler = jahrTicketZaehler + monatTicketZaehler;
25      jahrNutzungsZaehler = jahrNutzungsZaehler + monatNutzungsZaehler;
26      if ( jahrMax < monatMax )
27        jahrMax = monatMax;
28      end if
29      if ( jahrMin > monatMin )
30        jahrMin = monatMin;
31      end if
32      //Reset Monat:
33      monatTicketZaehler = 0;
34      monatNutzungsZaehler = 0;
35      monatMax = 0;
36      monat = td.Monat;
37      monatMin = GetMonatstage(monat);
38    end if
39    //Ticketauswertung:
40    monatTicketZaehler += 1;
41    monatNutzungsZaehler += td.NutzungsZaehler;
42    if ( monatMax < td.NutzungsZaehler )
43      monatMax = td.NutzungsZaehler;
44    end if
45    if ( monatMin > td.NutzungsZaehler )
46      monatMin = td.NutzungsZaehler;
47    end if
48  end foreach
++  //Monatsauswertung (letzter Monat):
++  WriteLn("Auswertung für Monat" + monat);
++  WriteLn(" Minimale Nutzung:" + monatMin);
++  WriteLn(" Maximale Nutzung:" + monatMax);
++  WriteLn(" Durchschnitt:" + monatNutzungsZaehler / monatTicketZaehler);
++  WriteLn(" Gesamtanzahl Tickets:" + monatTicketZaehler);
++  WriteLn();
++  //Monatsdaten für Jahresauswertung übernehmen (letzter Monat):
++  jahrTicketZaehler = jahrTicketZaehler + monatTicketZaehler;
++  jahrNutzungsZaehler = jahrNutzungsZaehler + monatNutzungsZaehler;
++  if ( jahrMax < monatMax )
++    jahrMax = monatMax;
++  end if
++  if ( jahrMin > monatMin )
++    jahrMin = monatMin;
++  end if
49  //Jahresauswertung:
50  WriteLn("Auswertung für Jahr" + td0.Jahr);
51  WriteLn(" Minimale Nutzung:" + jahrMin);
52  WriteLn(" Maximale Nutzung:" + jahrMax);
53  WriteLn(" Durchschnitt:" + jahrNutzungsZaehler / jahrTicketZaehler);
54  WriteLn(" Gesamtanzahl Tickets:" + jahrTicketZaehler);
55  WriteLn();
56 end TicketStatistik
```

ba) 2 Punkte

Jede Anweisung im Code muss mindestens einmal durchlaufen werden.

bb) 4 Punkte

Die Anweisungen innerhalb der if-Anweisungen zum Setzen der Minima werden nicht durchlaufen.

bc) 4 Punkte

Id	Monat	Jahr	Nutzungszaehler
0	Januar	2023	31
1	Januar	2023	31
2	Februar	2023	7

4. Aufgabe (25 Punkte)

a) 5 Punkte

```
SELECT COUNT(*) FROM Pflegearbeit WHERE YEAR(Datum_Soll) =2023 AND
Datum_Abschluss IS NULL
```

b) 8 Punkte

```
SELECT P.Datum_Soll AS Datum,
       WEEKDAY(P.Datum_Soll) AS Wochentag,
       T.Bezeichnung AS Tätigkeit
FROM Objekt AS O INNER JOIN Pflegearbeit AS P ON O.OID = P.OID
                 INNER JOIN Taetigkeit AS T ON T.TID = P.TID
WHERE Datum_Soll >= '19.06.2023' AND Datum_Soll <= '30.06.2023' AND
O.Bezeichnung = 'Außenanlage Nord'
ORDER BY Datum_Soll ASC
```

c) 8 Punkte

```
SELECT M.MID, M.Name, M.Vorname, (Count(MID_Ist) / 12) AS Durchschnitt
FROM Mitarbeiter AS M  LEFT JOIN Pflegearbeit AS P ON M.MID = P.MID_Ist
AND YEAR(Datum_Abschluss) = 2021
GROUP BY M.MID, M.Name, M.Vorname
```

d) 4 Punkte

```
CREATE USER Maier IDENTIFIED BY "5jk2T?"
GRANT SELECT ON Objekt TO Maier
```

Abschlussprüfung Sommer 2023

IHK

Belegsatz

Fachinformatiker/Fachinformatikerin
Anwendungsentwicklung (AO 2020)
1201

Teil 2 der Abschlussprüfung

Der Belegsatz ist Grundlage für beide Prüfungsbereiche.

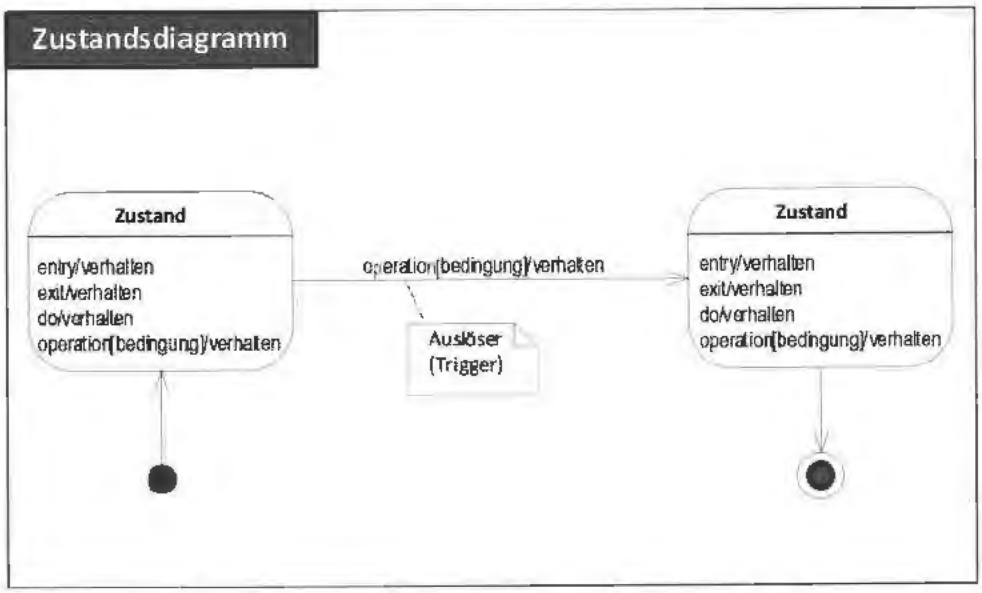
1. Planen eines Softwareproduktes
2. Entwicklung und Umsetzung von Algorithmen

	Seite
UML-Zustandsdiagramm	2
UML-Klassendiagramm	2
UML-Aktivitätsdiagramm	3
Beschreibung von Funktionen der Tabellenkalkulation	4
Struktogramm DIN 66261	5
SQL-Syntax (Auszug)	6/7

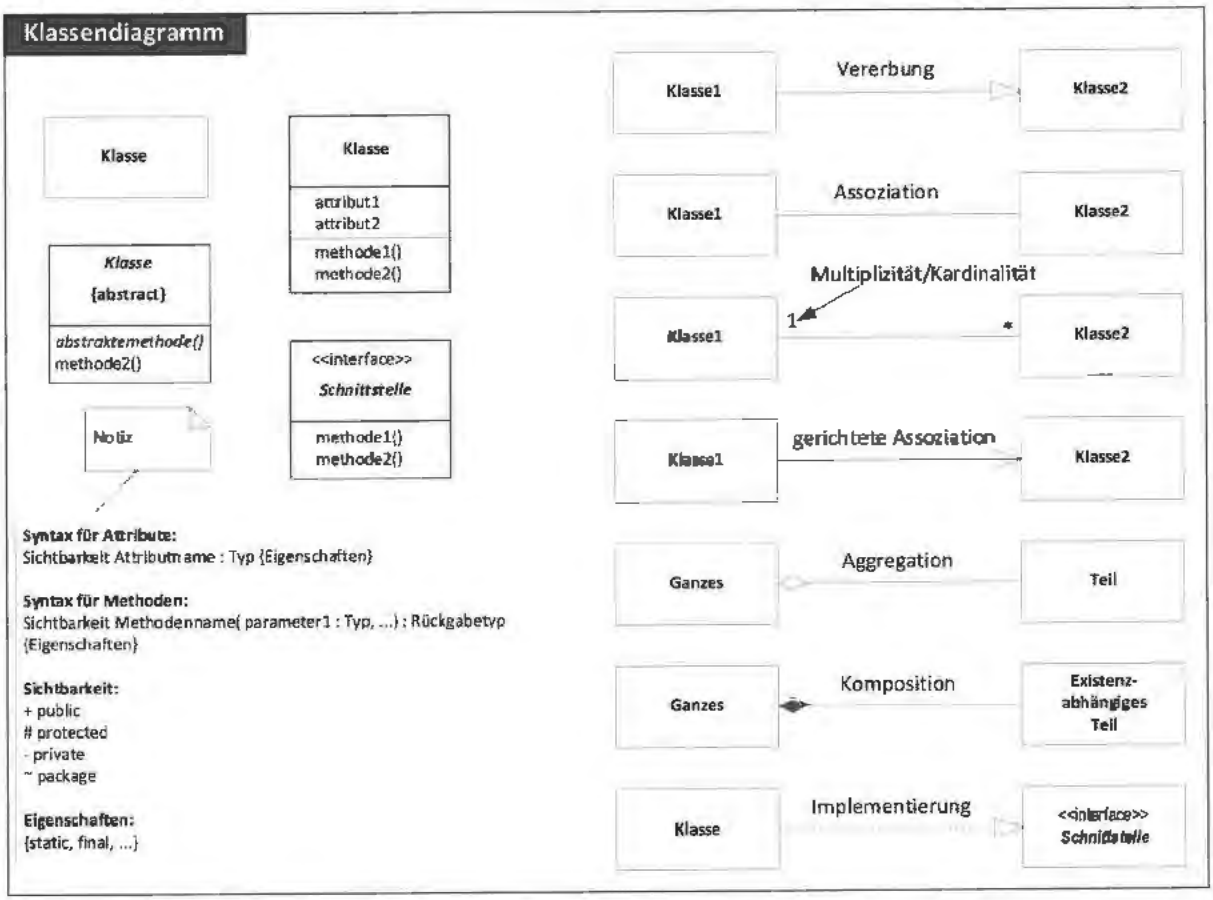
Gemeinsame Prüfungsaufgaben der Industrie- und Handelskammern. Dieser Aufgabensatz wurde von einem überregionalen Ausschuss, der entsprechend § 40 Berufsbildungsgesetz zusammengesetzt ist, beschlossen. Hinweis: Im Interesse einer besseren Lesbarkeit wird in der Aufgabenstellung und in den Angaben zur Aufgabenstellung nur die männliche Form (generisches Maskulinum) verwendet. Die verkürzte Sprachform beinhaltet keine Wertung und die gewählten männlichen Formulierungen gelten uneingeschränkt auch für die weiteren Geschlechter.

Die Vervielfältigung, Verbreitung und öffentliche Wiedergabe der Prüfungsaufgaben und Lösungen ist nicht gestattet. Zuwiderhandlungen werden zivil- und strafrechtlich (§§ 97 ff., 106 ff. UrhG) verfolgt. – © ZPA Nord-West 2023 – Alle Rechte vorbehalten!

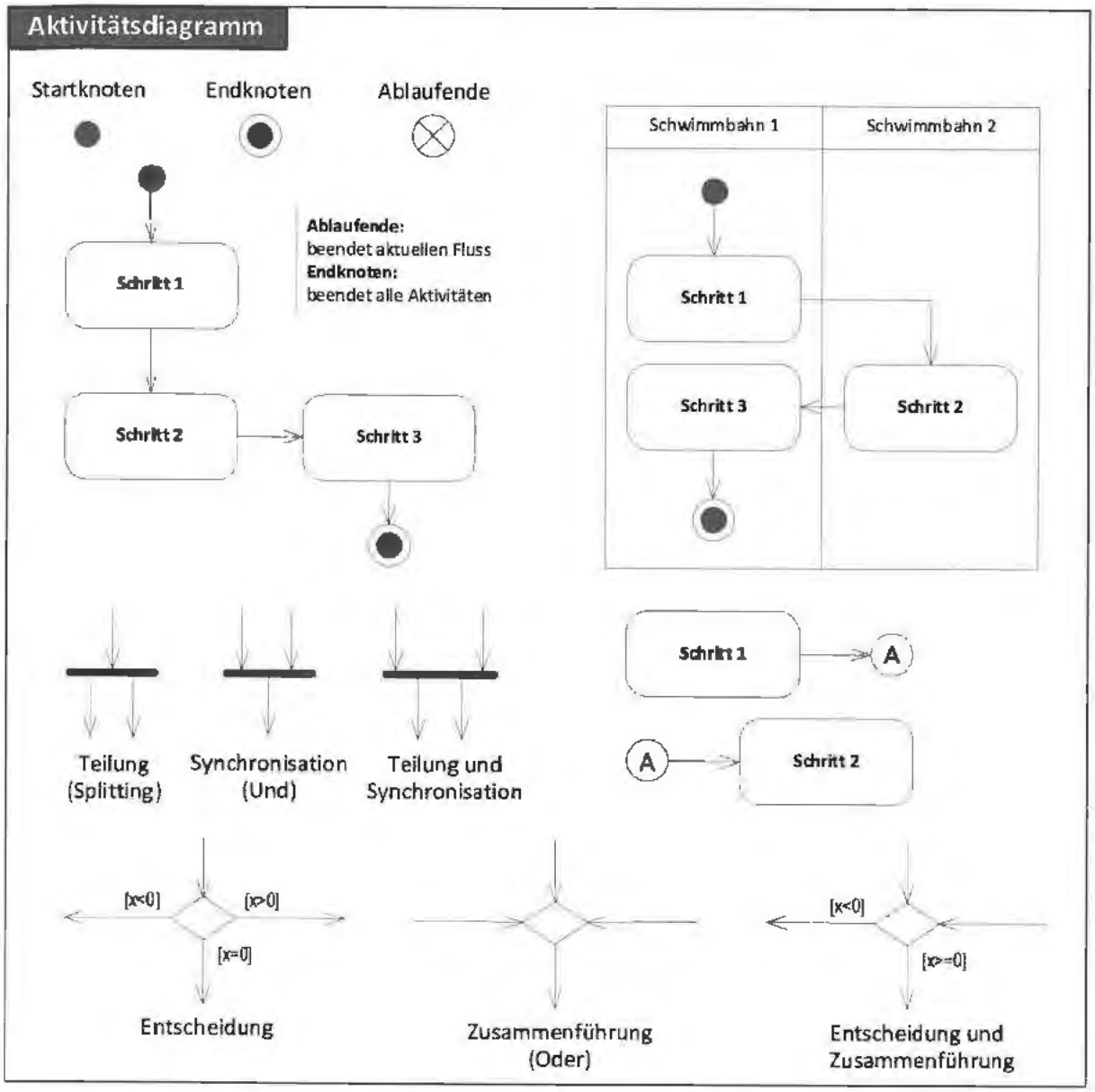
UML-Zustandsdiagramm



UML-Klassendiagramm



UML-Aktivitätsdiagramm



Beschreibung von Funktionen der Tabellenkalkulation

The function: VLOOKUP – Description
You can use the **VLOOKUP** function to search the first column of a range of cells, and then return a value from any cell on the same row of the range.
For example, suppose that you have a list of employees contained in the range A2:C10 and the employees' ID numbers are stored in the first column of the range and the employees' names are stored in the third column of the range.
If you know the employee's ID number, you can use the **VLOOKUP** function to return either the department or the name of that employee. To obtain the name of employee number 38, you can use the formula **=VLOOKUP(38; A2:C10; 3; FALSE)**. This formula searches for the value 38 in the first column of the range A2:C10, and then returns the value that is contained in the third column of the range and on the same row as the lookup. **FALSE** indicates you are searching for an exact match.

The function: IF – Description
The IF function returns one value if a condition you specify evaluates to TRUE, and another value if that condition evaluates to FALSE.
For example, the formula **=IF(A1>1; "Over 10";"10 or less")** returns "Over 10" if A1 is greater than 10, and "10 or less" if A1 is less than or equal to 10.

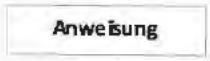
The function: WEEKDAY – Description
Returns the day of the week corresponding to a date. The day is given as an integer, ranging from 1 (Monday) to 7 (Sunday), by default.
For example, given the date 26.04.2023 in A1, the formula **=WEEKDAY(A1)** returns 3.

Range: An area of two or more cells.

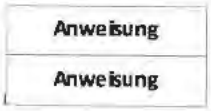
Quelle: https://support.microsoft.com/

Struktogramm DIN 66261

Struktogramm

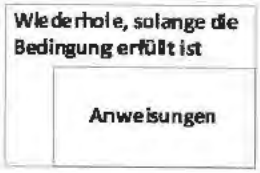


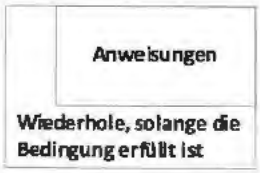
Verarbeitungsanweisung



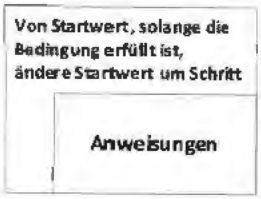
Sequenz (Folgestruktur), Anweisungen

Bedingte Verzweigung (einseitige/zweiseitige Auswahlstruktur)



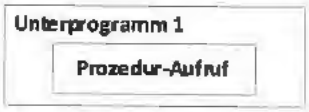


Wiederholstruktur (Schleife) **kopf**gesteuert und **fuß**gesteuert



Zählschleife

Fallauswahl (Mehrfach-verzweigung)



Unterprogramm

Wertzuweisungen werden durch := gekennzeichnet.

SQL-Syntax (Auszug)

Syntax	Beschreibung
Tabelle	
CREATE TABLE Tabellenname(Spaltenname < DATENTYP >, Primärschlüssel, Fremdschlüssel)	Erzeugt eine neue leere Tabelle mit der beschriebenen Struktur
ALTER TABLE Tabellenname **ADD COLUMN** Spaltenname Datentyp **DROP COLUMN** Spaltenname Datentyp **ADD FOREIGN KEY**(Spaltenname) **REFERENCES** Tabellenname(Primärschlüsselspaltenname)	Änderungen an einer Tabelle: Hinzufügen einer Spalte Entfernen einer Spalte Definiert eine Spalte als Fremdschlüssel
CHARACTER	Textdatentyp
DECIMAL	Numerischer Datentyp (Festkommazahl)
DOUBLE	Numerischer Datentyp (Doppelte Präzision)
INTEGER	Numerischer Datentyp (Ganzzahl)
DATE	Datum (Format DD.MM.YYYY)
PRIMARY KEY (Spaltenname)	Erstellung eines Primärschlüssels
FOREIGN KEY (Spaltenname) **REFERENCES** Tabellenname(Primärschlüsselspaltenname)	Erstellung einer Fremdschlüssel-Beziehung
DROP TABLE Tabellenname	Löscht eine Tabelle
Befehle, Klauseln, Attribute	
SELECT * | Spaltenname1 [, Spaltenname2, ...]	Wählt die Spalten einer oder mehrerer Tabellen, deren Inhalte in die Liste aufgenommen werden sollen; alle Spalten (*) oder die namentlich aufgeführten
FROM	Name der Tabelle oder Namen der Tabellen, aus denen die Daten der Ausgabe stammen sollen
SELECT ... **FROM** ... **(SELECT ...** **FROM ...** **WHERE ...) AS** tbl **WHERE** ...	Unterabfrage (subquery), die in eine äußere Abfrage eingebettet ist. Das Ergebnis der Unterabfrage wird wie eine Tabelle – hier mit Namen "tbl" – behandelt.
SELECT **DISTINCT**	Eliminiert Redundanzen, die in einer Tabellen auftreten können, Werte werden jeweils nur einmal angezeigt.
JOIN / INNER JOIN	Liefert nur die Datensätze zweier Tabellen, die gleiche Datenwerte enthelten
LEFT JOIN / LEFT OUTER JOIN	Liefert von der erstgenannten (linken) Tabelle alle Datensätze und von der zweiten Tabelle jene, deren Datenwerte mit denen der ersten Tabelle übereinstimmen
RIGHT JOIN / RIGHT OUTER JOIN	Liefert von der zweiten (rechten) Tabelle alle Datensätze und von der ersten Tabelle jene, deren Datenwerte mit denen der zweiten Tabelle übereinstimmen
WHERE	Bedingung, nach dar Datensätze ausgewählt werden sollen
WHERE EXISTS (subquery) **WHERE NOT EXISTS** (subquery)	Die Bedingungen EXISTS prüft, ob die Suchbedingung einer Unterabfrage mindestens eine Zeile zurückliefert. NOT EXIST negiert die Bedingung.
WHERE ... IN (subquery) **WHERE NOT... IN** (subquery)	Der Wert des Datenfelds ist in der auswählten Menge vorhanden. Der Wert des Datenfelds ist in der auswählten Menge nicht vorhanden.
GROUP BY Spaltenname1 [,Spaltenname2, ...]	Gruppierung (Aggregation) nach Inhalt des genannten Feldes
ORDER BY Spaltenname1 [,Spaltenname2, ...] **ASC | DESC**	Sortierung nach Inhalt des genannten Feldes oder der genannten Felder ASC: aufsteigend; DESC: absteigend

Syntax	Beschreibung
Datenmanipulation	
DELETE FROM Tabellenname	Löschen von Datensätzen in der genannten Tabelle
UPDATE Tabellenname **SET**	Aktualisiert Daten in Feldern einer Tabelle
INSERT INTO Tabellenname[(spalte1, spalte2, ...)] **VALUES** (Wert für Spalte 1 [, Wert für Spalte 2, ...]) oder **SELECT ... FROM ... WHERE**	Fügt Datensätze in die genannte Tabelle, die entweder mit festen Werten belegt oder Ergebnis eines SELECT-Befehls sind
Berechtigungen kontrollieren	
CREATE Benutzer | Rolle **IDENTIFIED BY** 'Passwort'	Erzeugt einen neuen Benutzer oder eine neue Rolle mit einem Passwort
GRANT Recht | Rolle **ON** *.* | Datenbank.* | Datenbank.Objekt **TO** Benutzer | Rolle [WITH GRANT OPTION]	Weist einem Benutzer oder einer Rolle ein Recht auf ein bestimmtes Datenbank-Objekt zu Weist einem Benutzer eine Rolle zu
REVOKE Rechte | Rollen **ON** *.* | Datenbank.* | Datenbank.Objekt **FROM** Benutzer | Rolle	Entzieht einem Benutzer oder einer Rolle ein Recht auf ein bestimmtes Datenbank-Objekt Entzieht einem Benutzer eine Rolle
Aggregatfunktionen	
AVG(Spaltenname)	Ermittelt das arithmetische Mittel aller Werte im angegebenen Feld
COUNT(Spaltenname | *)	Ermittelt die Anzahl der Datensätze mit Nicht-NULL-Werten im angegebenen Feld oder alle Datensätze der Tabelle (dann mit Operator *)
SUM(Spaltenname | Formel)	Ermittelt die Summe aller Werte im angegebenen Feld oder der Formelergebnisse
MIN(Spaltenname | Formel)	Ermittelt den kleinsten aller Werte im angegebenen Feld
MAX (Spaltenname | Formel)	Ermittelt den größten aller Werte im angegebenen Feld
Funktionen	
LEFT(Zeichenkette, Anzahlzeichen)	Liefert *Anzahlzeichen* der Zeichenkette von links.
RIGHT(Zeichenkette, Anzahlzeichen)	Liefert *Anzahlzeichen* der Zeichenkette von rechts.
CURRENT	Liefert das aktuelle Datum mit der aktuellen Uhrzeit
CONVERT(**time**,[DatumZeit])	Liefert die Uhrzeit aus einer DatumZeit-Angabe
DATE(Wert)	Wandelt einen Wert in ein Datum um
DAY(Datum)	Liefert den Tag des Monats aus dem angegebenen Datum
MONTH(Datum)	Liefert den Monat aus dem angegebenen Datum
TODAY	Liefert das aktuelle Datum
WEEKDAY(Datum)	Liefert den Tag der Woche aus dem angegebenen Datum als Text
YEAR(Datum)	Liefert das Jahr aus dem angegebenen Datum
DATEADD(Datumsteil, Intervall, Datum)	Fügt einem Datum ein Intervall (ausgedrückt in den unter Datumsteil angegebenen Einheiten) hinzu
DATEDIFF(Datumsteil, Anfangsdatum, Enddatum) Datumsteile: **DAY**, **MONTH**, **YEAR**	Liefert Enddatum-Startdatum (ausgedrückt in den unter Datumsteil angegebenen Einheiten)
Operatoren	
AND	Logisches UND
LIKE	Überprüfung von Text auf Gleichheit wenn Platzhalter ("regular expressions") eingesetzt werden.
NOT	Logische Negation
OR	Logisches ODER
IS NULL	Überprüfung auf NULL
=	Test auf Gleichheit
>, >=, <, <=, < >	Test auf Ungleichheit
*	Multiplikation
/	Division
+	Addition, positives Vorzeichen
-	Subtraktion, negatives Vorzeichen

Stand 2021-09-30